Title: Cooperative 3D Path Optimization (C3PO) Simulation

Authors: Blake Wall, Michael Lind

Abstract

Given a problem to get a group of unmanned vehicles from a starting location to a goal in 3 dimensions, we were able to do so without a home base for communication. Starting out with full map knowledge, the group would elect a leader, plan a path using Rapidly-Exploring Random Trees (RRTs), and move to the goal using Artificial Potential Field. The simulation was created in the MASON multi-agent simulation framework, and we were able to show that RRTs are a viable solution for path planning in 3 dimensions. We were also able to show how certain input parameters affect the ability to plan paths quickly.

Background

Path planning in 3 dimensions is significantly harder than in 2 dimensions, because the third dimension adds many more possible paths to the goal. It is almost impossible to find an exact solution to this problem in real time, and even using approximation algorithms, calculating such paths in real time is difficult [1]. The path planning algorithm used must then almost certainly be probabilistic and will not provide a guaranteed shortest path. The most appropriate algorithm that we found for this calculation was Rapidly-Exploring Random Trees (RRTs) [2].

RRTs work by iteratively picking a random point in space and adding a branch onto the existing tree that extends towards that point. The tree initially consists of only the root, which is the starting point for the path. As branches are added, they are checked to make sure they meet the requirements that the unmanned vehicle (UV) can traverse from one state to the next (i.e. the turning radius is not too small) and that the path from one state to the next does not cause the UV to crash into any obstacles. This is an important feature of RRTs in that they can build paths where the UV is guaranteed to be able to follow the path. When non-holonomic constraints are put on the system, it dramatically decreases the possible moves from one state to the next, and taking these constraints into account while building the path saves the effort of having to recalculate the path because of the inability to follow the path. Eventually a branch is added that extends to the goal. Once this happens, the branch can be followed back to the root; this is the path returned from the RRT algorithm. It is important to note that the random point in space is chosen before the branch on which to add it. If the branch is chosen first, the tree tends to search space that has already been explored. RRTs have the nice property that they tend to explore

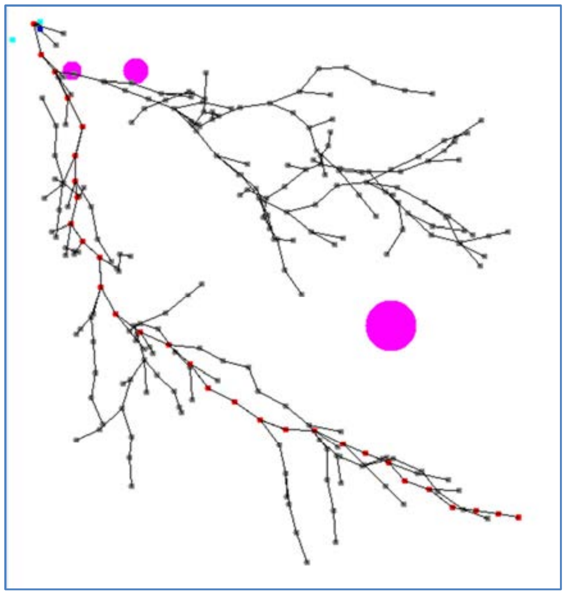


Figure 1: Path Planning using RRT algorithm

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **10 NOV 2015**	2. REPORT TYPE **Summary**	3. DATES COVERED **01 OCT 2014 - 31 SEP 2015**

4. TITLE AND SUBTITLE **Cooperative 3D Path Optimization (C3PO)**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) **Blake Wall Michael Lind**	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Space and Naval Warfare System Center Atlantic 1 Innovation Dr, Hanahan, SC 29410**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Given a problem to get a group of unmanned vehicles from a starting location to a goal in 3 dimensions, we were able to do so without a home base for communication. Starting out with full map knowledge, the group would elect a leader, plan a path using Rapidly-Exploring Random Trees (RRTs), and move to the goal using Artificial Potential Field. The simulation was created in the MASON multi-agent simulation framework, and we were able to show that RRTs are a viable solution for path planning in 3 dimensions. We were also able to show how certain input parameters affect the ability to plan paths quickly.

15. SUBJECT TERMS
Cooperative 3D Path Optimization Planning dimension swarm RRT artificial potential field robot unmanned vehicle

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**SAR**	**6**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

open spaces. Figure 1 shows an example RRT being built, and the red points would be returned as the path from start to the goal location.

MASON Simulation Framework

The MASON framework [3] is a discrete event framework written in Java that allows for rapid development of multi-agent simulations. It follows the Model-View-Controller paradigm. It can produce logs, track changes to variables over time, and produce graphics. It is easy to control parameters, even allowing for manually changing them during runtime. Simulations can be started, paused, saved to disk, and resumed at a later date. Graphics can also be turned off for faster simulations; this approach was taken after our design of experiments to calculate potentially troublesome or interesting relationships between specific input parameters and output parameters.

The simulations themselves were programed as follows: All UVs are assumed to be point objects with a heading. Movement would consist of moving along the heading and being able to turn the heading (in any direction) over time as allowed by the minimum turning radius. All obstacles are assumed to be point objects with different radii. Coming into contact with an obstacle's surface was assumed to be a crash. UVs could also come into contact with one another; this was assumed to be a different type of crash. Both types of crashes were captured.

Each UV was given full map knowledge before the simulation. However, location of other UVs was only able to be obtained by the other UVs broadcasting their locations. Only point to point communication was allowed and this was bounded by a maximum communication distance (variable over different experiments). Communication was considered to be imperfect and any partial message was dropped; this communication error rate was also variable over the experiments. Leader determination must occur locally within the agents themselves; no external decision maker is available to the UVs. Simple maps were used with minimal obstacles to prove the viability of RRTs; harder maps would have tested more of the shortcomings of movement by Artificial Potential Field (APF).

Analytical Methodology

In order to learn about this complex system, our analysis is accomplished in three phases: Designing simulations (aka "experiments"), running each simulation, and then evaluating different metamodels by best-fitting simulation results.

1 - Design of experiments

Our simulation contains 25 input factors and 8 output factors (listed in Appendix A). The authors utilized the SEED Center for Data Farming template for data design [4]. The purpose was to design a set of experiments that was Nearly Orthogonal and Balanced (NOB) which provides the framework for finding correlations between input and output factors while minimizing the overall number of design points. We were able to search for first-order interactions between inputs, as well as quadratic effects. The worksheet generated 512 such experiments over the full range of input values in each category (input ranges were chosen either arbitrarily or with domain knowledge based on research into the UV field).

2 – Ran simulations with replications

We simulated these experiments using MASON and captured the results. Each simulation was run until it met a Stopping Condition, which defined colloquially are “Mission Complete”, or “Ran out of time.”

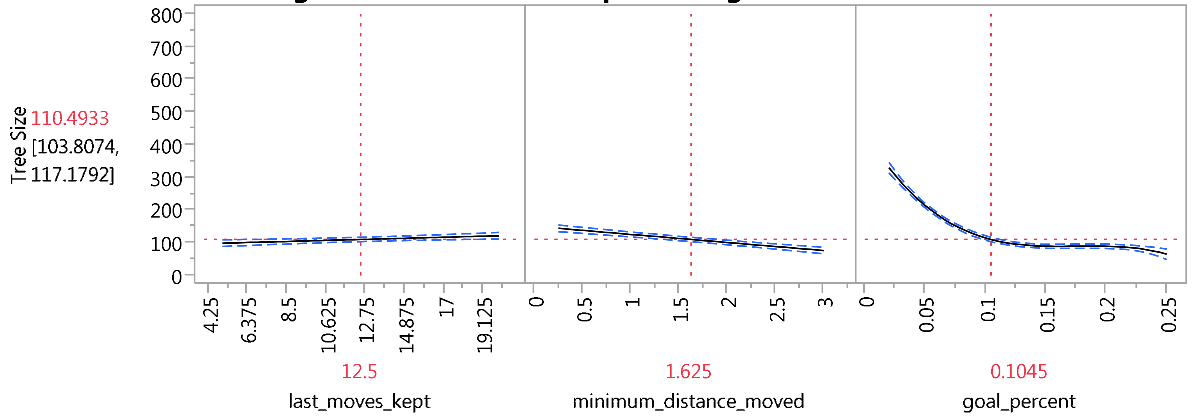
3 – Evaluated meta-modes

These results were then imported into JMP Pro [4], where we were able to find interesting relationships between input and output factors. Significant relationships are discussed below.

Findings

The most significant factor that goes into finding a path quickly is the GOAL_PERCENT. This number represents probability that the “random” point in space added to the RRT represents the goal. The higher this percent, the more often the tree tries to grow directly towards the goal. Results show that if goal_percent is too small (below 10%), a higher Tree Size will be produced.

Variable Profiler of Significant Variables on predicting Tree Size



This graphic shows the influence of goal_percent on Tree Size compared to the other statistically significant variables.

It is our hypothesis that as the maps get harder, having the GOAL_PERCENT be too large would increase the tree size dramatically; this should be tested in later experiments.

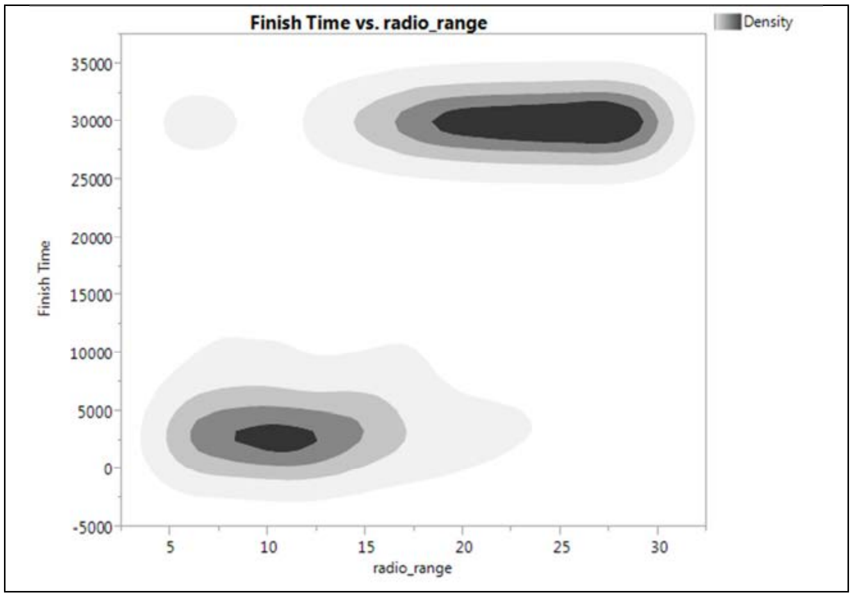
In our experiments, UVs were completely able to avoid each other. No UV crashes were reported in any of the experiments. Some experiments showed higher likelihood for UVs being within a “warning range,” but no input factors seemed to correlate with this output. However, there were sometimes obstacle crashes. The most significant feature that determined obstacle crashes was the virtual force applied by the obstacle in the APF algorithm. This is both intuitive and fairly obvious as to why that would lead to more crashes.

The number of paths calculated was another metric captured. One of the features of the APF algorithm is that it can get stuck in local minima where the force applied to it from all directions is essentially summed to 0. When this happens, the UV must find another path to the goal and this requires recalculation of the path. The way for a UV to know whether or not it is stuck is to determine how much it has moved over some amount of time. This calculation involves keeping a certain number of previous moves and having some minimum distance that must have been moved over that time period.

We've determined that the significant factors that can predict "Number of Paths Calculated" are: Last_moves_kept, minimum_disntance_moved, and uv_max_velocity.

More investigation is required, but we believe these results are related to following phenomenon: as UVs incorrectly assume they are in a local minimum when in fact, given their maximum velocity and the number of moves kept, they are simply unable to move the required minimum distance in time.

The last significant finding was that the radio range was the biggest factor in determining the time for all UVs to finish. As the radio range increased past a certain limit, UV missions almost always failed.



We believe this is a by-product of the simulation itself in that UVs only considered themselves to have reached the goal if the center point between all known UVs reached the goal. As the radio range increases, the UVs are able to communicate with those that were stuck/lost/crashed at significant distances away from the goal. This causes the UVs that reached the goal to consider themselves to not have finished either. Therefore, we've identified that knowledge of distance from other UVs is critical for mission success.

Recommendations and Future Work

Since GOAL_PERCENT is most significant factor for determining the tree size, it might be possible to parameterize it and tune it depending on specific map properties (number of obstacles, percent free space, etc.). It may also be possible to tune the maximum branch size for the tree to get longer branches. It makes sense that in open space this would lead to smaller (by number of nodes) trees. For future work, the authors would like to test out tuning these parameters on different maps. The authors would especially like to try starting with very high maximum branch size and gradually make it smaller as suitable paths can't be found.

The authors would also like to try this set of algorithms with more realistic movement, specifically quadcopter movement. Taking quadcopter limitations into consideration when planning the path with RRTs would be a viable next step for the development of this algorithm set for UV movement.

The authors would also like to use this set of algorithms when the UVs do not know the full map and obstacles are instead discoverable. This has been previously used in 2 dimensions with success, so 3 dimensions should be possible.

Works Cited

[1] Z. K. a. B. M. C. Goerzen, "A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance," *Journal of Intelligent and Robotic Systems,* pp. 65-100, 2010.

[2] S. M. Lavelle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," 1998.

[3] L. e. al., "Multi Agent Simulator Of Networks," [Online]. Available: https://cs.gmu.edu/~eclab/projects/mason/. [Accessed 22 September 2015].

[4] Vieira, Jr., H., Sanchez, S. M., Kienitz, K.H., Belderrain, M. C. N. 2012. Conducting trade-off- analyses via simulation: Efficient nearly orthogonal nearly balanced mixed design. Working paper, Operations Research Department, Naval Postgraduate School, Monterey, CA.

[5] S. Institute, "JMP Pro," [Online]. Available: http://www.jmp.com/en_us/software/jmp-pro.html. [Accessed 22 September 2015].

Appendix A

Input factors:	Output factors:
• radio_range • children_timeout • alive_timeout • time_per_move • send_info_delta • clock_tick • uv_mass • uv_max_velocity • uv_centripetal_force • uv_max_acceleration • last_moves_kept • minimum_distance_moved • goal_radius • sink_radius • goal_percent • obstacle_push_range • obstacle_push_constant • obstacle_push_type • uv_push_range • uv_push_constant • uv_push_type • uv_pull_range • uv_pull_constant • force_mask • coms_loss	• Time spent in the warning zone for any obstacle • Time spent crashing into any obstacle • Time spent in the warning zone for any UV • Time spent crashing into any UV • Number of path calculations • Number of simulation steps taken to get to the goal • Number of UVs that found the goal • Average tree size (by number of nodes)